- INHERITANCE (상속) –

Inheritance: Mechanism in which a child object (class) acquires all the properties and behaviours (methods and fields) of a parent object (class).

```
Class ChildClass extends ParentClass {
```

- A subclass can have only **one superclass.** Multiple inheritances are only allowed through interfaces
- A subclass inherits all the members **(fields, methods, and nested classes)** from its superclass.
- Constructors are not members, so they are not inherited by subclasses.
- **Private member inheritance:** A subclass does not inherit the private members of its parent class. However, if the superclass has public or protected methods (like getters and setters) for accessing its private fields, these can also be used by the subclass.

1. INTRODUCTION

Object class: The supreme ancestor of all classes (top of the heirachy of classes) that has no ancestor.

- It automatically inherits 11 default Object class methods: toString(), equals(Object obj), hashCode(), ...

Why inheritance is needed:

1. Helps avoid making the same members from scratch
2. Second, various objects (types) can be grouped into one object (type) through inheritance
   - Example: A person can breathe and walk. An animal can breathe and crawl. Abstraction can unite humans and animals under living things

2. OVERRIDING

Overriding: Saving multiple methods of the same name **in another class**. Often referred as polymorphism as the ability to be displayed in more than one form

- A method with the same name can be overloaded with different functions
- If a child class overrides a parent class method, the program immediately and solely executes the child class's overridden method (parent class cannot be used)

3. SUPER

Super: Reference variable that is used in the Child's class to identify objects of the parent class.

- When the method of the parent class cannot be used because it has been overridden by a childclass, the method can still be used using the 'super' keyword
- [super.]: A reference variable that is used to refer parent class **objects**. It calls parent class' variables and methods.
- [super ()]: A reference variable that is used to refer parent class constructors.

```
Class Parent {
        int number = 3;
Parent() {
        System.out.println("Parent object");
```

```
Class Child Extends Parent {
        int number = 2;
        Child() {
                System.out.println(Child object");
        }
        Void print () {
        int number = 1;
                System.out.println(number);
                System.out.println(this.number);
                System.out.println(super.number);
```

```
Public class SuperEx {

  Public static void main(String[] args) {

Child child = new Child();
Child.print();
```

```
}
}
```

```
Parent object
Child object
1
2
3
```

1: Looks for closest value of variable ‘number’
2: Printing the instance variable of the object ‘child’ in the ‘Child’ class
3: Looks for value of variable ‘number’ the parent object).